CS 1110 Spring 2022, Assignment 4: Nick and Norah's Recursive Playlist*

People can make playlists[1], or *mixes*, by selecting individual songs and/or by combining other mixes. That means we can think of mixes as recursive data structures!

The files you'll need are in this zip file:[2]
http://www.cs.cornell.edu/courses/cs1110/2022sp/assignments/assignment4/a4_skeleton.zip . a4_test.py contains test cases for the functions you'll complete in file a4.py.

```
Mix 'CS1110 picks' with contents:
  Mix "Chris's list" with contents:
    Song 'Something About Us'
  Mix "Prof. Bracy's list" with contents:
    Song 'Alles Neu'
    Song "Wavin' Flag"
    Song 'Wish You Were Here'
    Song 'Top of the World'
    Song 'We Are the Dinosaurs'
    Song 'Der Glühwurm Julius'
  Mix "Lifan's list" with contents:
    Song 'Desire is a Trap'
    Song 'NOTHING 3V3R CHAN935'
  Mix "Gary's list" with contents:
    Song 'Anomalie bleue'
    Song 'Liability'
    Song 'PPP'
    Song 'Veridis Quo'
    Song 'Goodbye To A World'
    Song 'Mercy'
    Song 'Aruarian Dance'
    Song 'Good News'
  Mix "Prof. Lee's list" with contents:
    Song 'Broken Arrow'
    Song 'Divorce Song'
    Song 'Nana Party (Chuang 2021 live version)'
    Mix 'LCD Soundsystem' with contents:
      Song 'Dance Yrself Clean'
      Song 'Losing My Edge'
```

Figure 1: The structure of Mix `a4_test.cs1110picks`. The main Mix is made up of Mixes from CS1110 course-staff members Chris, Prof. Bracy, Lifan, Gary, and Prof. Lee.

Contents

*Authors: Lillian Lee. Thanks to CS1110 staffer Jasmine Herrera for suggestion the class name Mix.

[1] We henceforth abandon the term "playlist" to avoid confusion with Python lists.

[2] If you are using a Windows computer: when you download a zip file, it is represented in File Explorer as a folder with a little zip on it. That is not a real folder, which is why you have may have had trouble saving changes to files in it! So, first create a new folder, such as "a4-real". Double-click on that zip pseudo-folder. Then, move the contents of the pseudo-folder to folder a4-real.

1 Rules

1.1 New rules

1. A major goal of this assignment is practice with recursion. Hence, *we reserve the right to assign no credit for implementations that aren't fundamentally based on recursion in the way we ask for*, even if the code fulfills the specification.

2. You *are* allowed to use for-loops[3] as long as recursion is the fundamental basis of your implementation.

3. You *are* allowed to use negative list indices and, as a means to de-duplicate certain lists, the set() function as described in Section 6.

4. You may *not* use any Python that was not introduced in class or the associated materials by the time this assignment was released.[4]

1.2 Previous rules that still apply

See Sections 1.1-1.3 of Assignment 1[5], Sections 3.1-3.2 of Assignment 2[6], and Sections 3.1.3 and 3.2 (pre-submission checklist) of Assignment 3[7].

2 Timeline

1. If you are partnering,[8] **form your group on CMS *before* submitting anything**. This links your submission "portals".

2. By 2pm Ithaca time on Fri Apr 1, submit your *partial* progress on CMS.[9] It is OK if you haven't finished working yet.[10]

[3] But not while-loops, if you even know what those are.

[4] **In particular, you are not allowed to use list comprehensions, map(), or filter(), even if for some reason you have heard of these things before.**

[5] `https://www.cs.cornell.edu/courses/cs1110/2022sp/assignments/assignment1/a1.pdf`

[6] `https://www.cs.cornell.edu/courses/cs1110/2022sp/assignments/assignment2/a2.pdf`

[7] `https://www.cs.cornell.edu/courses/cs1110/2022sp/assignments/assignment3/a3.pdf`

[8] Reminder: Both parties need to act on CMS in order for the grouping to take effect. See the "How to form a group" instructions at `https://www.cs.cornell.edu/courses/cs1110/2022sp/resources/cms.html` .

[9] And, as usual, perform steps 1-3 in the "Updating, verifying, and documenting assignment submission" section of `https://www.cs.cornell.edu/courses/cs1110/2022sp/resources/cms.html`

[10] The 2pm checkpoints provide you a chance to alert us if any problems arise, and for partners to become aware if something has gone wrong with the CMS grouping process (via us emailing people who haven't yet submitted — if your partner submitted but you got the warning email, that means CMS doesn't think you've grouped!). Since we'll send a reminder around 2pm to all those who haven't submitted yet, do not expect that we will accept work that doesn't make it onto CMS on time, for whatever reason. There are no so-called "slipdays" and there is no "you get to submit late at the price of a late penalty" policy. Of course, if some special circumstances arise, contact the instructor(s) immediately.

3. By **11:59pm (Ithaca time) on Fri Apr 1**, make your final submission, and do steps 1-3 in the "Updating, verifying, and documenting assignment submission" section of https://www.cs.cornell.edu/courses/cs1110/2022sp/resources/cms.html

3 Classes Song and Mix

File a4_classes.py defines these classes, but you don't need to look at it. Here's all you need to know.

1. A Song object has two attributes, `title` and `artist`, both non-empty strings.

   A call of the form `Song(t, a)` creates a new Song with `title` attribute set to `t` and `artist` attribute set to `a`.

2. A Mix object has two attributes, `title` and `contents`. `title` is a non-empty string. `contents` is a non-empty list, where each element is either a Song (not None) or a Mix. There should be no cycles/loops in the containment relationships of any of the contents of the Mixes reachable from a given Mix's contents.[11]

   A call of the form `Mix(t, c)` creates a new Mix with `title` attribute set to `t` and `contents` attribute set to `c`.

3. If you want to check whether an item `x` is a Song or a Mix, we'd like you to transition from using `type(x)== Song` (ditto Mix) to the following expression instead:

   `isinstance(x, Song)`

   (ditto for Mix).[12]

In this assignment, for both types of objects, you can use the sorted() function or the sort() list method whenever we ask you to produce sorted lists.

3.1 Example: a non-recursive Mix

Here is code, drawn from a4_test.py, that creates a small Mix titled "why the big suit"[13] whose `contents` attribute is a list of five Songs. Each Song is stored in a variable whose name starts with "s_" and then has the initials of the Song title.

```
s_tmbtp = Song("This must be the place", "Talking Heads")
s_pk    = Song("Psycho Killer", "Talking Heads")
s_gib   = Song("Girlfriend is better", "Talking Heads")
s_oial  = Song("Once in a lifetime", "Talking Heads")
s_bdth  = Song("Burning Down the House", "Talking Heads")

th = Mix("why the big suit", [s_tmbtp, s_pk, s_gib, s_oial, s_bdth])
```

Check your understanding: do you see that the following expressions evaluate to the given values? If not, consider drawing folder diagrams or using Python Tutor.

Expression	Value
`th`	A Mix.
`th.contents`	A list of 5 Song objects
`th.contents[3].title`	string "Once in a lifetime"

3.2 Example: a recursive Mix

This code creates a Mix whose `contents` attribute contains both Songs and the Mix created previously.

```
s_tmbtp2 = Song("This Must be the Place", "The Lumineers")
s_bdth2  = Song("Burning Down the House", "Tom Jones and the Cardigans")

# This Mix contains both a sub-Mix and two individual songs
th2 = Mix("Talking Heads Songs and Covers", [th, s_tmbtp2, s_bdth2])
```

[11] A Mix object is always considered "reachable" from itself.
[12] Why? We may reuse your A4 code for later assignments that will do some more sophisticated things with classes and subclasses.
[13] This is a reference to lore regarding the band Talking Heads.

```
Mix 'Eccentrica' with contents:  # `weird'
  Mix 'Songs in the key of Falsetto' with contents: # `high`
    Song 'This Town Ain't Big Enough for the Both of Us' # s_ttabeftbou
    Song 'Redbone' # s_r
    Song 'Bennie and the Jets' # s_batj
    Song 'Bohemian Rhapsody' # s_br
  Mix 'Recs from M' with contents: # `m_recs`
    Mix 'Talking Heads Songs and Covers' with contents: # `th2`
      Mix 'why the big suit' with contents: # `th`
        Song 'This must be the place' # s_tmbtp
        Song 'Psycho Killer' # s_pk
        Song 'Girlfriend is better' # s_gib
        Song 'Once in a lifetime' # s_oial
        Song 'Burning Down the House' # s_bdth
      Song 'This Must be the Place' # s_tmbtp2
      Song 'Burning Down the House' # s_bdth2
    Mix 'LCD Soundsystem' with contents: # `lcd'
      Song 'Dance Yrself Clean' # s_dyc
      Song 'Losing My Edge' # s_lme
```

Figure 2: The structure of the Mix `a4_test.weird`, as indicated by applying a4_classes.pprint_mix(). We've added comments indicating variables we've stored the component Songs/Mixes in, for your reference.

Check your understanding: do you see that the following expressions evaluate to the given values?

Expression	Value
`th2.contents[1]`	A Song with `title` "This Must be the Place" and `artist` "The Lumineers".
`th2.contents[0]`	A Mix with `title` "why the big suit"
`th2.contents[0].contents[3].title`	string "Once in a lifetime"

3.3 Example of a more-nested Mix and a recursive function

Other code in a4_test.py creates the Mix stored in variable `weird`. We wrote a "pretty-printing" function pprint_mix() on Mixes for you in a4_classes.py. The output uses indentation to indicate inclusion. The output of that function on `weird` is shown in Figure 2.

And, for your enjoyment, we provide Figure 1, depicting a Mix of picks from the CS1110 staff. Section A lists all the Songs and Mixes created in a4_test.py, for your reference.

4 Task: `songs_in()`

One might want to pull out a plain old non-nested list of the Songs in a Mix.

```
def songs_in(m):
    """Returns: sorted list of all *distinct* Songs that can be reached by
    tracing through the contents of `m`.

    Two Songs count as non-distinct if they have the same title and artist.

    `m` itself should remain unchanged.

    Examples:
        Suppose we had a Mix whose contents was [a4_test.llee, a4_test.weird].

        The `llee` Mix contains, using the variable names from a4_test.py,
           s_ba, s_ds, s_np, s_dyc, s_lme .
        The `weird` Mix can be worked out to contain
            s_ttabeftbou, s_r, s_batj, s_br, s_tmbtp, s_pk, s_gib, s_oial,
```

```
        s_bdth, s_tmbtp2, s_bdth2, s_dyc, s_lme
    There are two Songs in common: s_dyc and s_lme.

    The result should be (since the default sort order for Songs is by
    artist, then title) the list of the following Songs, in this order:

        s_r:    Song 'Redbone' by Childish Gambino
        s_batj: Song 'Bennie and the Jets' by Elton John
        s_dyc:  Song 'Dance Yrself Clean' by LCD Soundsystem
        s_lme:  'Losing My Edge' by LCD Soundsystem
        s_ds:   'Divorce Song' by Liz Phair
        s_np    'Nana Party (Chuang 2021 live version)' by OJM [et al.]
        s_br:   'Bohemian Rhapsody' by Queen
        s_ba:   'Broken Arrow' by Robbie Robertson
        s_ttabeftbou: 'This Town Ain't Big Enough for the Both of Us' by Sparks
        s_bdth: 'Burning Down the House' by Talking Heads
        s_gib   'Girlfriend is better' by Talking Heads
        s_oial: 'Once in a lifetime' by Talking Heads
        s_pk:   'Psycho Killer' by Talking Heads
        s_tmbtp:'This must be the place' by Talking Heads
        s_tmbtp2:'This Must be the Place' by The Lumineers
        s_bdth2 :'Burning Down the House' by Tom Jones and the Cardigans

    Precondition: m is a Mix (not None).
    """
    assert isinstance(m, Mix), "songs_in: input not a Mix. It is: "+repr(m)

    pass # STUDENTS: implement this, using recursion effectively and explicitly.
         #
         # For the purposes of this assignment, you may not use map().
         # You *can't* use set() to remove duplicate Songs because of Python
         #  implementation issues (Song objects aren't "hashable").
         #
```

This task requires creating a list without duplicates. It would be good practice to figure out how to do so on your own, but we give you an example of how to do this in a4_test.py:[14]

```
182         result = a4.basic_mixes(m)
183         result_no_dups = [] # Construct deduplicated version
184         for r in result:
185             if r not in result_no_dups:
186                 result_no_dups.append(r)
```

5 Task: `basic_mixes()`

If we have a Mix built out of layers and layers of pre-existing Mixes, we might be interested in what were the "earliest" or most basic sub-Mixes that the Mix was drawn from.

```
def basic_mixes(m):
    """Returns: a sorted list of the Mixes that are "reachable" from `m`
    and whose `contents` list consists entirely of Songs.

    In other words, in the list it returns are the reachable sub-Mixes
    (or sub-sub-, or sub-sub-sub-...Mixes) that are not themselves recursive
    in structure.
```

[14] Python prevents us from using the "list(set())" trick introduced in Section 6 because the set() function doesn't work for our Song and Mix objects.

```
    `m` itself can be in the returned list.

    Duplicates in the returned list are OK.

    Example: for the `weird` Mix in a4_test,
     * the "reachable" Mixes are:
       `weird` itself, `high`, `m_recs`, `th2`, `lcd`, `th`.
     * The Mixes that should be in the returned list are:
      `th`, `lcd`, and `high` (sorted by title).

    Precondition:
        m: a Mix (not None)
    """
    assert isinstance(m, Mix), "basic_mixes: input not a Mix. It is "+repr(m)
    pass # STUDENTS: implement this as an explicitly and effectively recursive
         # function.
         #
         # For this exercise, you may not use map(), even if you know what it is.
         # You can use `break` to terminate a for-loop early, but you don't
         #  need to (or need to know what it is).
```

6 Task: `mixes_with()`

Given a Song and a list of Mixes, we might want to know which of the Mixes contain that Song. For example, if you have excellent taste and are wondering which of your friends do too, you might check to see which of your friends have Mixes that include a specific "signal" Song.

```
def mixes_with(s, mlist):
    """Returns: sorted list of *titles* of Mixes in `mlist` that Song `s` is in
        (either directly or through indirect chain of containment).

    Preconditions:
        s: a Song (not None)
        mlist: a non-empty list of Mixes

    Example:
     if `s` were a4_test.s_m (that is, the Song with title 'Mercy'),
     and `mlist` were [a4_test.cs1110picks, a4_test.sad, a4_test.th],
     then the output would be:
       ['CS1110 picks', "Gary's list", 'Sad songs']

     * The first two are a result of the Song titled 'Mercy' being in the Mix
      titled "Gary's list", which is in the Mix titled 'cs1110 picks'.

     * The third comes from the Song being in the Mix titled 'Sad songs'.

     * The Song 'Mercy' isn't in the Mix titled 'why the big suit', so no
     more entries are in the output.
    """
    assert isinstance(s, Song) and isinstance(mlist, list)
    for item in mlist:
        assert isinstance(item, Mix)

    pass # STUDENTS: implement this as an explicitly and effectively recursive
         # function.
         # For this function, you may not use map() or songs_in(), nor create
```

```
        # any lists of Songs. We want you to use fresh ideas, not
        # re-use the "list-flattening" technique from songs_in().
        #
        # You *are* allowed to use list(set()) to remove duplicates.
```

The list returned by this function should not have any duplicates in its string. For a list of strings (or other "hashable" objects) `x`, `list(set(x))` returns a deduplicated version of `x`.

7 Advice

1. For debugging print statements with Mixes, use our a4_classes.pprint_mix() function.
2. a4_classes.pprint_mix() is a complete example of a recursive function on Mixes, so you may want to take it as a model.[15]
3. Only implement a little bit at a time and test incessantly. It's normal to let many tests fail for code you haven't implemented yet, as long as what you *are* working on is getting closer to functioning.[16]

   You don't want an uncaught bug early one messing up a lot of things downstream.
4. Many bugs are caused by unintentionally changing the semantics of a variable. Pick informative variable names and/or comment what your intents are. Make sure you update variable values correctly when the situation changes.
5. Once one gets the key insight, recursive solutions tend to be quite elegant and compact. If you find yourself making modification after modification but not making progress, and/or if your code seems to be getting longer and longer, *save your work*, but try starting over, on a blank sheet of paper.

   A fresh start can sometimes work wonders.
6. Section 13.10 of the text ("...especially if you are working on a hard bug") is good advice.

A Alphabetized list of all Song and Mix variables in a4_test.py

In case it helps you read/understand the test cases, here is an alphabetically-sorted list of assignment statements creating relevant variables, Songs first, then Mixes.

```
s_ab   = Song("Anomalie bleue", "L'Imperatrice")
s_ad   = Song("Aruarian Dance", "Nujabes")
s_an   = Song("Alles Neu", "Peter Fox")
s_ba = Song("Broken Arrow", "Robbie Robertson")
s_batj = Song("Bennie and the Jets", "Elton John")
s_bdth  = Song("Burning Down the House", "Talking Heads")
s_bdth2  = Song("Burning Down the House", "Tom Jones and the Cardigans")
s_br   = Song("Bohemian Rhapsody", "Queen")
s_dgj  = Song("Der Glühwurm Julius", "Ute Rink")
s_diat = Song("Desire is a Trap", "Bladee & Ecco2k")
s_ds = Song("Divorce Song", "Liz Phair")
s_dyc = Song("Dance Yrself Clean", "LCD Soundsystem")
s_gib   = Song("Girlfriend is better", "Talking Heads")
s_gn   = Song("Good News", "Mac Miller")
s_gtaw = Song("Goodbye To A World", "Porter Robinson")
s_l    = Song("Liability", "Lorde")
s_lme = Song("Losing My Edge", "LCD Soundsystem")
s_m    = Song("Mercy", "Max Richter")
s_n3c = Song("NOTHING 3V3R CHAN935", "Brainiac")
s_np = Song("Nana Party (Chuang 2021 live version)", "OJM, Johan Gustafson, Fredrik Haggstam, Sebastian Lundberg, and W
```

[15] Trying to actually re-implement it would be a great exercise, but doing so is trickier than any of the required tasks in this assignment.
[16] (Ha.)

```
s_oial  = Song("Once in a lifetime", "Talking Heads")
s_p    = Song("PPP", "Beach House")
s_pk    = Song("Psycho Killer", "Talking Heads")
s_r    = Song("Redbone", "Childish Gambino")
s_sau = Song("Something About Us", "Daft Punk")
s_tmbtp = Song("This must be the place", "Talking Heads")
s_tmbtp2 = Song("This Must be the Place", "The Lumineers")
s_totw = Song("Top of the World", "Carpenters")
s_ttabeftbou=Song("This Town Ain't Big Enough for the Both of Us", "Sparks")
s_vq   = Song("Veridis Quo", "Daft Punk")
s_watd = Song("We Are the Dinosaurs", "The Laurie Berkner Band")
s_wf   = Song("Wavin' Flag", "K'naan")
s_wywh = Song("Wish You Were Here", "Pink Floyd")



bracy = Mix("Prof. Bracy's list", [s_an, s_wf, s_wywh, s_totw, s_watd, s_dgj])
chris = Mix("Chris's list", [s_sau])
cs1110picks = Mix("CS1110 picks", [chris, bracy, lifan, gary, llee])
elec  = Mix("electronic music", [s_vq, s_ab, s_gtaw])
gary  = Mix("Gary's list", [s_ab, s_l, s_p, s_vq, s_gtaw, s_m, s_ad, s_gn])
hhr   = Mix("hip-hop/rap", [s_ad, s_gn])
high = Mix("Songs in the key of Falsetto", [s_ttabeftbou, s_r, s_batj, s_br])
instr = Mix("instrumentals", [s_vq, s_m, s_ad])
lcd = Mix("LCD Soundsystem", [s_dyc, s_lme])
lifan = Mix("Lifan's list", [s_diat, s_n3c])
llee= Mix("Prof. Lee's list", [s_ba, s_ds, s_np, lcd])
m_recs = Mix("Recs from M", [th2, lcd])
sad   = Mix("Sad songs", [s_l, s_p, s_gtaw, s_m, s_dyc, s_wywh, s_ds])
th = Mix("why the big suit", [s_tmbtp, s_pk, s_gib, s_oial, s_bdth])
th2 = Mix("Talking Heads Songs and Covers", [th, s_tmbtp2, s_bdth2])
weird = Mix("Eccentrica",[high, m_recs])
```